ALIBABA CLOUD

Alibaba Cloud

Data Transmission Service

Best Practices

Document Version: 20220712

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Switch workloads to the destination database

This topic describes how to switch your workloads to the destination database and prepare a rollback solution. This allows you to minimize the negative impact of data migration on your business.

Prerequisites

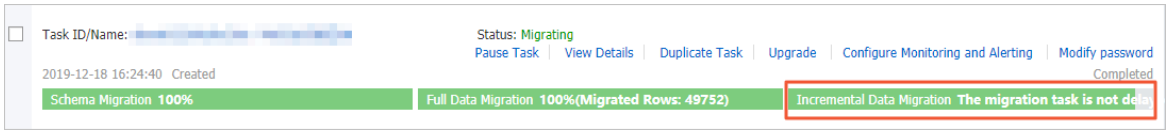
A data migration task is configured and it is in the **Migrating** or **Completed** state. For more information, see Overview of data migration scenarios.

Precautions

- We recommend that you switch workloads to the destination database during off-peak hours to minimize the negative impact. Before you switch workloads to the destination database, you must stop writing data to the source database and suspend the business.
- We recommend that you create and authorize a database account for data migration. This allows you to distinguish session information and improve data security.

Procedure

1. Wait until the task progress bar shows **Incremental Data Migration** and **The migration task is not delayed** or a delay time of less than 5 seconds.



> **Note** If you do not select **Incremental Data Migration** when you configure the data migration task, the task progress bar does not show **Incremental Data Migration**. After data is migrated, the migration task automatically ends. In this case, you must suspend the business and stop writing data to the source database before you run the data migration task. Skip to Step 5 and proceed.

2. Suspend the business and stop writing data to the source database.
3. Log on to the source database and run the following statements based on the database type to view session information. Make sure that no new sessions are used for write operations.

> **Note** You can view the processes or sessions between DTS and the source database by running the preceding statements.

◉ MySQL ○ SQL Server ○ Oracle ○ PostgreSQL ○ Redis ○ MongoDB

```
show processlist;
```

```
select * from sys.dm_exec_connections;
```

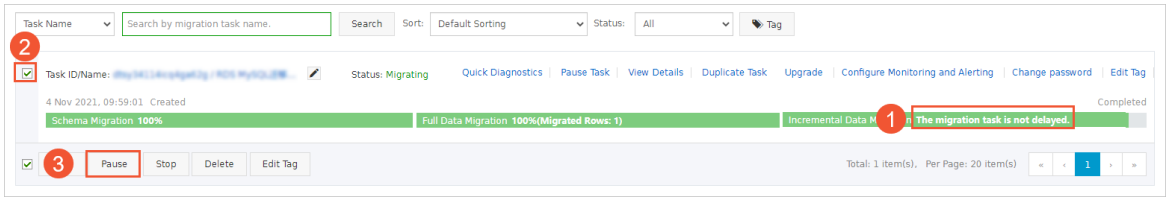
```
select sid,serial#,username,program,machine,status from v$session;
```

```
select * from pg_stat_activity;
```

```
CLIENT LIST
```

```
use admin
db.runCommand({currentOp: 1, $all:[{"active" : true}]})
```

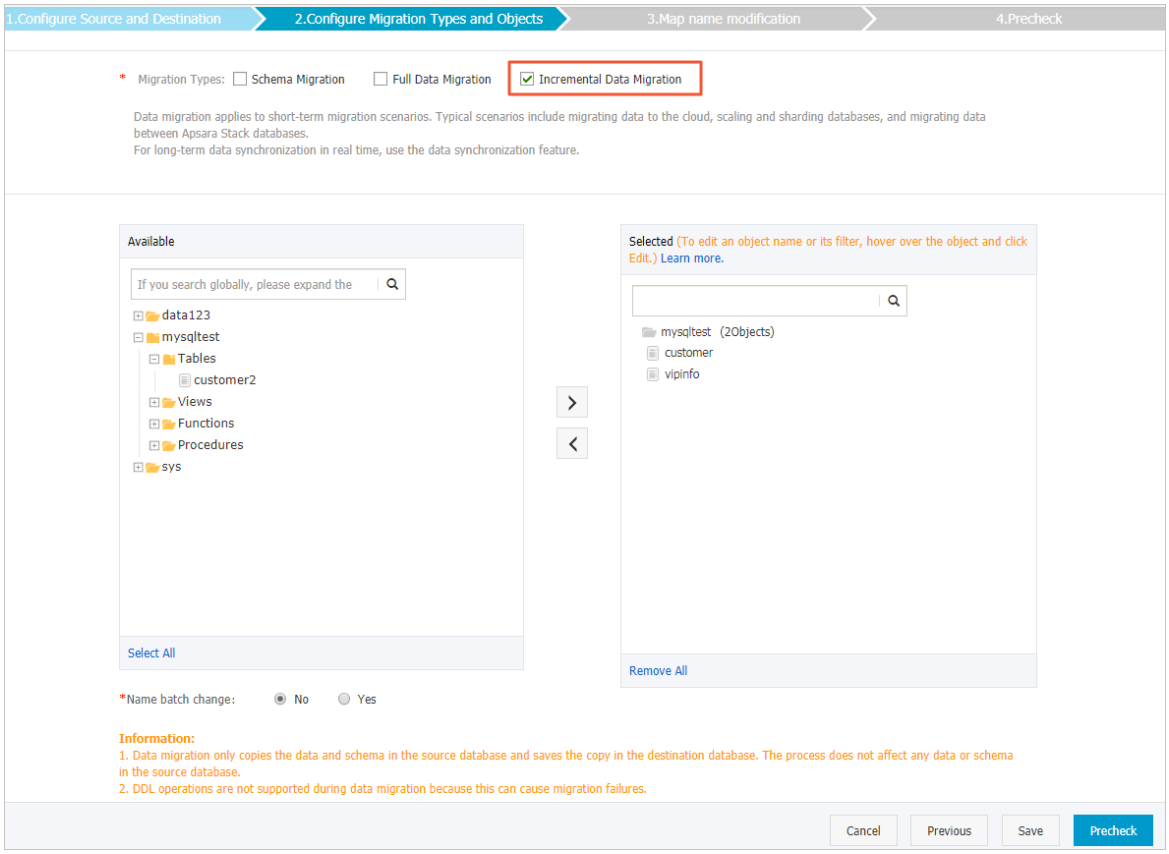
4. After the status of **incremental data migration** changes to **The migration task is not delayed**, wait for 1 minute or longer, and then manually stop the migration task.



5. During service interruption, remove the limit on writing data to the source database.
6. Create and start a task to migrate incremental data generated in the destination database to the source database. The migration task created in this step provides a rollback solution. If an error occurs in the destination database, you can switch workloads to the source database.

   For example, data is migrated from a self-managed MySQL database to an ApsaraDB RDS for MySQL instance. To create a task in the opposite direction, see Migrate data from an ApsaraDB RDS for MySQL instance to a self-managed MySQL database (select only **Incremental Data Migration**).

> **Warning** When you configure a data migration task in the opposite direction, you must select only **Incremental Data Migration** in the **Configure Migration Types and Objects** step. Then, you must select the database or table to be migrated back to the source database.



7. Verify that the data of the source and destination databases is consistent, switch workloads to the destination database, and then resume your business.

   > **Warning** If you do not verify data consistency before you switch workloads and resume your business, data between the source and destination databases may become inconsistent.

8. After you run the task in the opposite direction, incremental data generated in the destination database is migrated back to the source database in real time. If the business fails, you can switch workloads back to the source database.

What to do next

After you switch workloads to the destination database and test all the business-related features, you can stop the task in the opposite direction. For more information, see Stop a data migration task.

> **Warning** The database accounts that are used for data migration have the read and write permissions. After data is migrated, you must delete the accounts or revoke the write permission to ensure security.

FAQ

- Q: What can I do if an error occurs after I switch workloads to the destination database?

  A: If an error occurs, you can switch workloads back to the source database. After you run the task in the opposite direction, incremental data generated in the destination database is migrated back to the source database in real time.

- Q: How can I ensure data consistency in the source database if I am unable to switch workloads to

the destination database?

A: You can back up the source database before you switch workloads.

- Q: What can I do if data is written to the source database due to a misoperation after I switch workloads to the destination database?

  A: You can compare data of the source and destination database through data verification and manually change data to ensure consistency.

2.Use triggers and functions to implement incremental DDL migration for PostgreSQL databases

Before you use Data Transmission Service (DTS) to migrate data between PostgreSQL databases, you can create a function and a trigger in the source database. The function and trigger obtain the data definition language (DDL) information of the source database. During incremental data migration, DTS migrates DDL operations to the destination database.

Prerequisites

- The source database must meet the following requirements:
  - If the source database is a self-managed PostgreSQL database, the database version must be V9.4 or later.
  - If the source database is an ApsaraDB RDS for PostgreSQL instance, the version of the ApsaraDB RDS for PostgreSQL instance must be V10 or later.
    - ApsaraDB RDS for PostgreSQL V9.4 does not support event triggers.
    - The kernel versions of ApsaraDB RDS for PostgreSQL V10, V11, and V12 must be 20201130 or later.
    - The kernel versions of ApsaraDB RDS for PostgreSQL V13 must be 20210228 or later.

    > **Note** For more information about how to upgrade the kernel version of ApsaraDB RDS for PostgreSQL, see Update the minor engine version of an ApsaraDB RDS for PostgreSQL instance.

- A data migration task was created after October 1, 2020.

Context

When you use DTS to migrate data between PostgreSQL databases, DTS synchronizes only data manipulation language (DML) operations during incremental data migration. DML operations include INSERT, DELETE, and UPDATE. DTS does not synchronize DDL operations during incremental data migration.

To synchronize DDL operations, you can create a trigger and a function to obtain the DDL information of the source database. During incremental data migration, DTS migrates DDL operations to the destination database.

> **Note** Only the following DDL operations can be synchronized: CREATE TABLE, DROP TABLE, and ALTER TABLE. The ALTER TABLE operation includes RENAME TABLE, ADD COLUMN, and DROP COLUMN.

Procedure

Warning If you need to migrate incremental data from multiple databases, repeat Steps 2 to 5 for each database.

1. Log on to the source PostgreSQL database. For more information, see Connect to an ApsaraDB RDS for PostgreSQL instance or psql.
2. Switch to the source database.

   Note The psql tool is used in this example. You can run the `\c <Database name>` command to switch to the source database, for example, `\c dtststdata`.

3. Execute the following statements to create a table that stores the DDL information:

```
CREATE TABLE public.dts_ddl_command
(
    ddl_text text COLLATE pg_catalog."default",
   id bigserial primary key,
   event text COLLATE pg_catalog."default",
   tag text COLLATE pg_catalog."default",
   username character varying COLLATE pg_catalog."default",
   database character varying COLLATE pg_catalog."default",
   schema character varying COLLATE pg_catalog."default",
   object_type character varying COLLATE pg_catalog."default",
   object_name character varying COLLATE pg_catalog."default",
   client_address character varying COLLATE pg_catalog."default",
   client_port integer,
   event_time timestamp with time zone,
   txid_current character varying(128) COLLATE pg_catalog."default",
   message text COLLATE pg_catalog."default"
);
```

4. Execute the following statements to create a function that obtains the DDL information:

```
CREATE FUNCTION public.dts_capture_ddl()
    RETURNS event_trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF SECURITY DEFINER
AS $BODY$
  declare ddl_text text;
  declare max_rows int := 10000;
  declare current_rows int;
  declare pg_version_95 int := 90500;
  declare pg_version_10 int := 100000;
  declare current_version int;
  declare object_id varchar;
  declare alter_table varchar;
  declare record_object record;
  declare message text;
  declare pub RECORD;
begin
  select current_query() into ddl_text;
  if TG_TAG = 'CREATE TABLE' then -- ALTER TABLE schema.TABLE REPLICA IDENTITY FULL;
    show server_version_num into current_version;
```

```
    show server_version_num into current_version;
    if current_version >= pg_version_95 then
      for record_object in (select * from pg_event_trigger_ddl_commands()) loop
        if record_object.command_tag = 'CREATE TABLE' then
          object_id := record_object.object_identity;
        end if;
      end loop;
    else
      select btrim(substring(ddl_text from '[ \t\r\n\v\f]*[c|C][r|R][e|E][a|A][t|T][e|E
][ \t\r\n\v\f]*.*[ \t\r\n\v\f]*[t|T][a|A][b|B][l|L][e|E][ \t\r\n\v\f]+(.*)\(.*'),' \t\r
\n\v\f') into object_id;
    end if;
    if object_id = '' or object_id is null then
      message := 'CREATE TABLE, but ddl_text=' || ddl_text || ', current_query=' || cur
rent_query();
    else
      alter_table := 'ALTER TABLE ' || object_id || ' REPLICA IDENTITY FULL';
      message := 'alter_sql=' || alter_table;
      execute alter_table;
    end if;
    if current_version >= pg_version_10 then
      for pub in (select * from pg_publication where pubname like 'dts_sync_%') loop
        raise notice 'pubname=%',pub.pubname;
        BEGIN
          execute 'alter publication ' || pub.pubname || ' add table ' || object_id;
        EXCEPTION WHEN OTHERS THEN
        END;
      end loop;
    end if;
  end if;
  insert into public.dts_ddl_command(id,event,tag,username,database,schema,object_type,
object_name,client_address,client_port,event_time,ddl_text,txid_current,message)
  values (default,TG_EVENT,TG_TAG,current_user,current_database(),current_schema,'','',
inet_client_addr(),inet_client_port(),current_timestamp,ddl_text,cast(TXID_CURRENT() as
varchar(16)),message);
  select count(id) into current_rows from public.dts_ddl_command;
  if current_rows > max_rows then
    delete from public.dts_ddl_command where id in (select min(id) from public.dts_ddl_
command);
  end if;
end
$BODY$;
```

5. Change the owner of the function to the account that is used to connect to the source database, for example, postgresql.

```
ALTER FUNCTION public.dts_capture_ddl()
    OWNER TO postgres;
```

6. Execute the following statements to create a global event trigger:

```
CREATE EVENT TRIGGER dts_intercept_ddl ON ddl_command_end
EXECUTE PROCEDURE public.dts_capture_ddl();
```

What's next

Configure a data migration task. For more information, see the following topics:

- Migrate incremental data from a self-managed PostgreSQL database (in PostgreSQL 10.0 or an earlier version) to an ApsaraDB RDS for PostgreSQL instance
- Migrate incremental data from a self-managed PostgreSQL database (version 10.1 to 13) to an ApsaraDB RDS for PostgreSQL instance

> **Note** After the data migration task is released, you must log on to the source PostgreSQL database and execute the following statements to delete the trigger and function.

```
drop EVENT trigger dts_intercept_ddl;
drop function public.dts_capture_ddl();
drop table public.dts_ddl_command;
```

3.Disable slow query log to improve migration performance

This topic describes how to disable slow query log for a destination ApsaraDB RDS for MySQL instance to improve the performance of data migration. We recommend that you disable slow query log if the instance has a low specification (less than two CPU cores) and you want to migrate a large volume of data.

Prerequisites

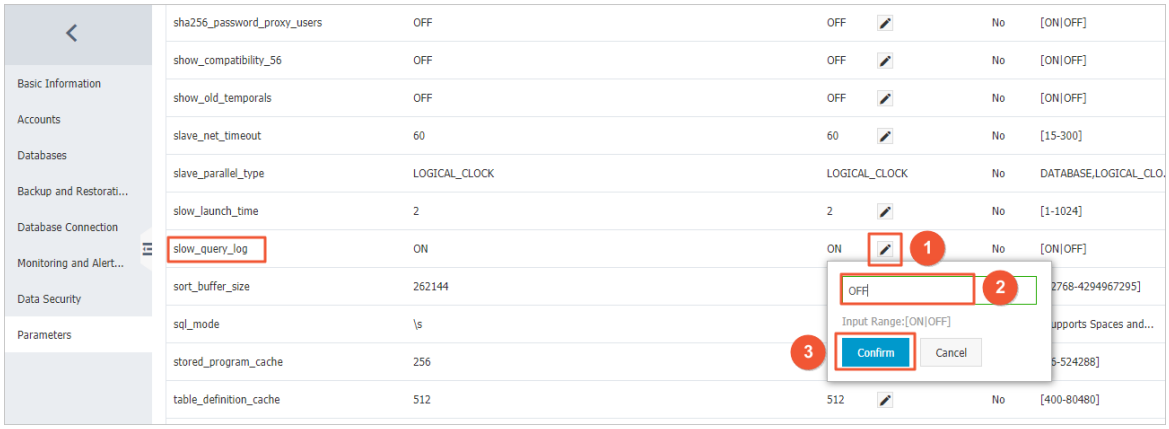
The database version of ApsaraDB RDS for MySQL is 5.6, 5.7, or 8.0.

Context

When migrating data to the destination ApsaraDB RDS for MySQL instance, DTS generates a large number of log entries in the instance. The log collection program of the instance scans and queries slow log tables. This increases the CPU load of the instance and compromises the performance of data migration.

Procedure

1. Log on to the ApsaraDB for RDS console.
2. In the upper-left corner of the page, select the region where the instance resides.
3. Find the instance and click the instance ID.
4. In the left-side navigation pane, click **Parameters**.
5. Find the **slow_query_log** parameter, click the edit icon, set the parameter value to **OFF**, and then click Confirm.



> **Note** After the data migration task is completed, you can set the value of the **Slow_query_log** parameter to **ON**.

4.Change the character set of an ApsaraDB RDS for MySQL instance

You may need to change the character set of a table (for example, from GBK to UTF8mb4) based on your business requirements. However, an ALTER statement that is used to change the character set of a large table will have negative impacts on your business.

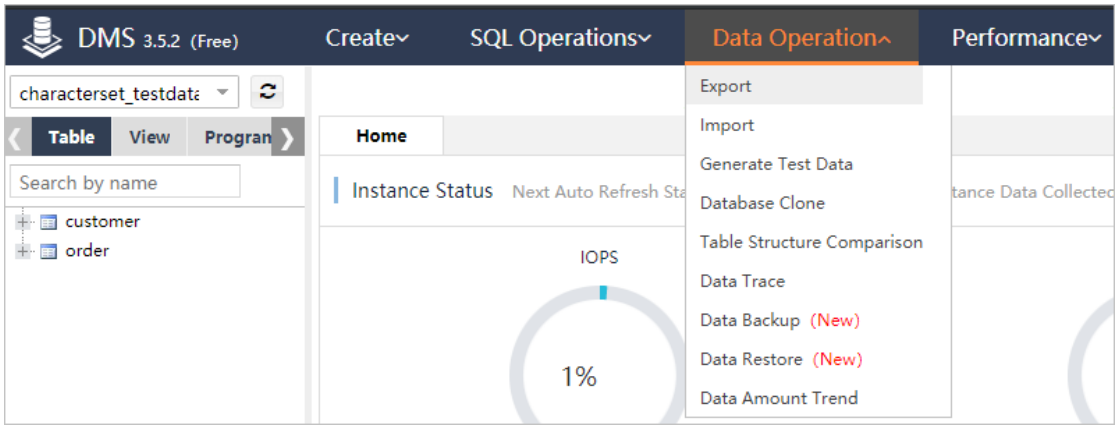
This topic describes how to change the character set without affecting your business. You can create a table schema in the destination instance based on the new character set, and then use DTS to migrate table data to the destination instance. To further ensure service continuity, you can prepare a rollback solution or a dual-write solution.

Precautions

- Before you change the character set, ensure that your business system and SQL statements are compatible with the new character set and the features of the business system will not be affected.
- During full data migration, DTS uses read/write resources of the source and destination instances. This may increase the database load. Before you migrate data, evaluate the performance of the source and destination instances. We recommend that you migrate data during off-peak hours.
- The source instance must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, duplicate data may exist in the destination instance.

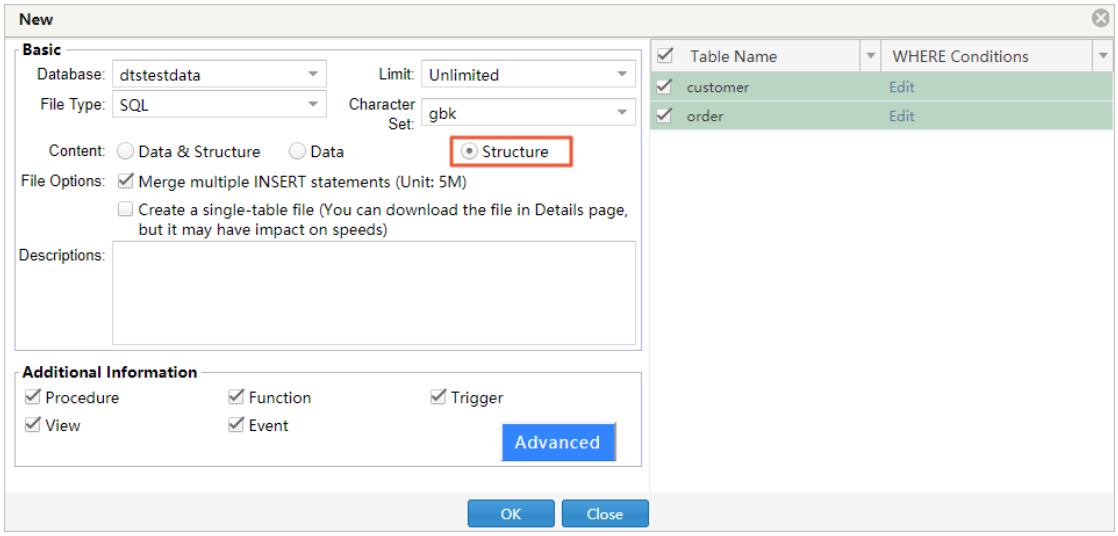
Step 1: Import the table schema from the source instance to the destination instance

1. Export the script of the target table schema from the source instance. The target table schema refers to the schema of the table whose character set you want to change.
   i. Log on to the source instance by using Data Management (DMS). For more information, see Use DMS to log on to an ApsaraDB RDS for MySQL instance.
   ii. In the top navigation bar, choose **Data Operation > Export**.



   iii. On the page that appears, choose **New > Export Database**.

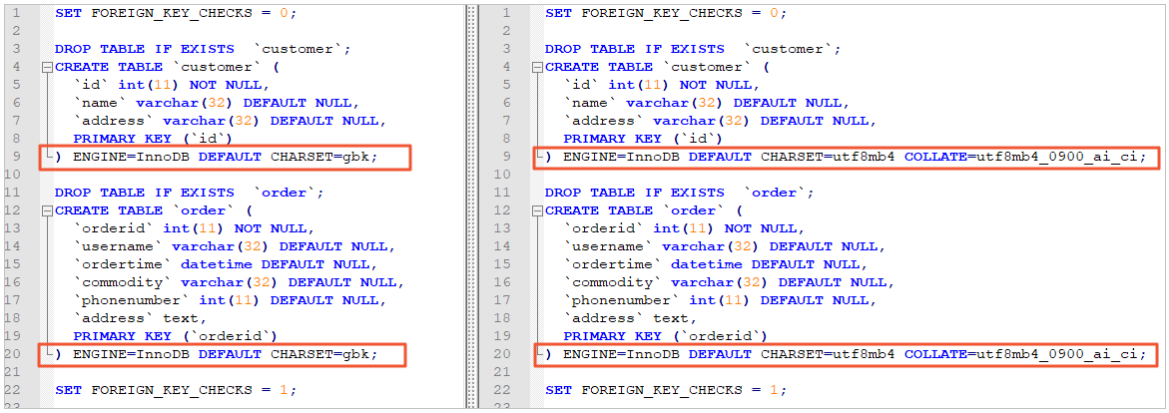
iv. In the dialog box that appears, specify the parameters.



> **Note** Select the destination database, select **Structure** for the **Content** parameter, and specify the other parameters based on your business requirements.

v. Click **OK**. In the dialog box that appears, click **YES**.

2. Decompress the exported file, and change the character set in the script of the table schema.



```
SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS `customer`;
CREATE TABLE `customer` (
  `id` int(11) NOT NULL,
  `name` varchar(32) DEFAULT NULL,
  `address` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;

DROP TABLE IF EXISTS `order`;
CREATE TABLE `order` (
  `orderid` int(11) NOT NULL,
  `username` varchar(32) DEFAULT NULL,
  `ordertime` datetime DEFAULT NULL,
  `commodity` varchar(32) DEFAULT NULL,
  `phonenumber` int(11) DEFAULT NULL,
  `address` text,
  PRIMARY KEY (`orderid`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk;

SET FOREIGN_KEY_CHECKS = 1;
```

```
SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS `customer`;
CREATE TABLE `customer` (
  `id` int(11) NOT NULL,
  `name` varchar(32) DEFAULT NULL,
  `address` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

DROP TABLE IF EXISTS `order`;
CREATE TABLE `order` (
  `orderid` int(11) NOT NULL,
  `username` varchar(32) DEFAULT NULL,
  `ordertime` datetime DEFAULT NULL,
  `commodity` varchar(32) DEFAULT NULL,
  `phonenumber` int(11) DEFAULT NULL,
  `address` text,
  PRIMARY KEY (`orderid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

SET FOREIGN_KEY_CHECKS = 1;
```

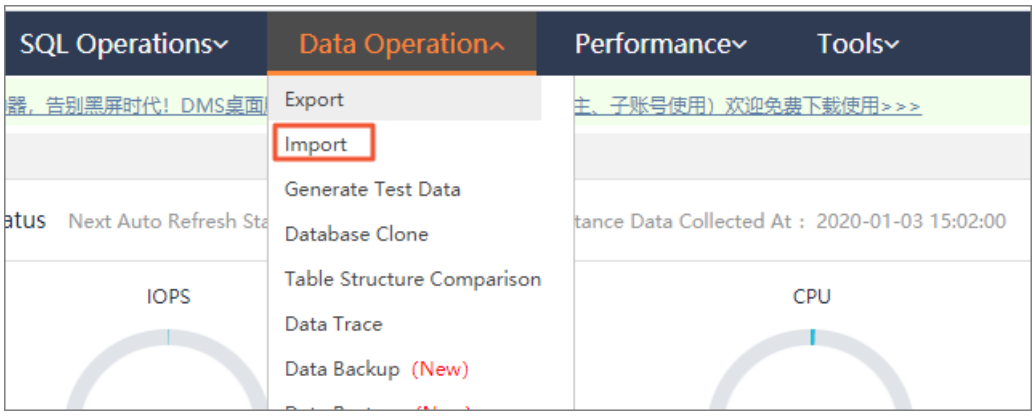
> **Note** The example shown in this figure changes the GBK character set to UTF8mb4.

3. Create an ApsaraDB RDS for MySQL instance of the same type as the source instance. The new instance is the destination instance. For more information about how to create an ApsaraDB RDS for MySQL instance, see Create an ApsaraDB RDS for MySQL instance.

4. Import the edited script of the table schema to the new ApsaraDB RDS for MySQL instance.

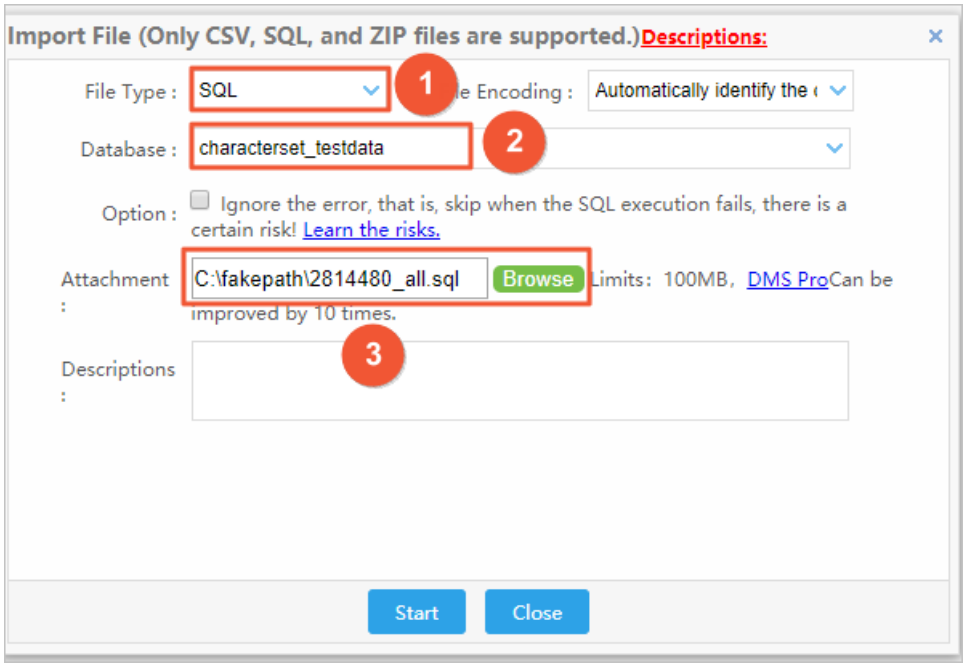
i. Log on to the destination ApsaraDB RDS for MySQL instance by using DMS. For more information, see Use DMS to log on to an ApsaraDB RDS for MySQL instance.

ii. In the top navigation bar, choose **Data Operation > Import**.



iii. On the page that appears, click **New Task**.

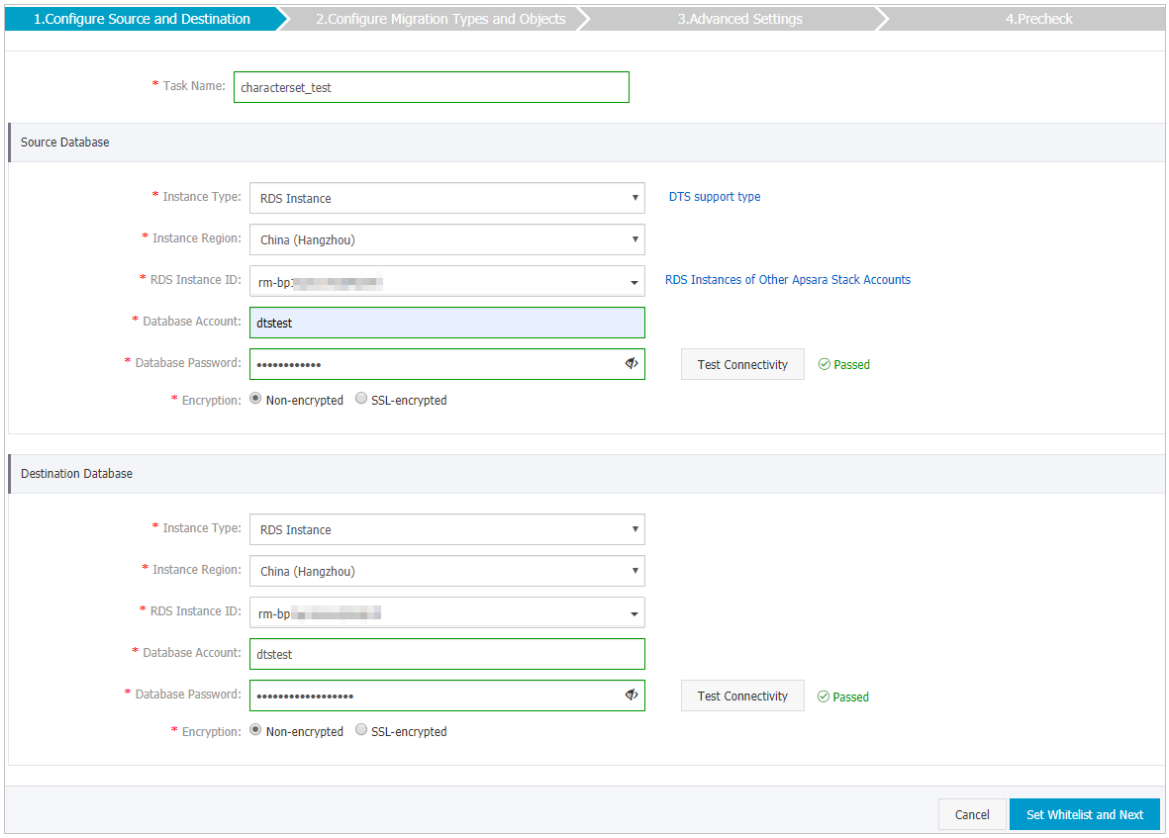
iv. In the dialog box that appears, specify the parameters, as shown in the following figure. Then, click **Start**.



> **Note** After the table schema is imported, you can execute the `show create table <Table name>;` statement to verify the new character set of the table.

Step 2: Migrate table data from the source instance to the destination instance

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. At the top of the **Migration Tasks** page, select the region where the destination cluster resides.
4. In the upper-right corner of the page, click **Create Migration Task**.
5. Configure the source and destination databases of the data migration task.



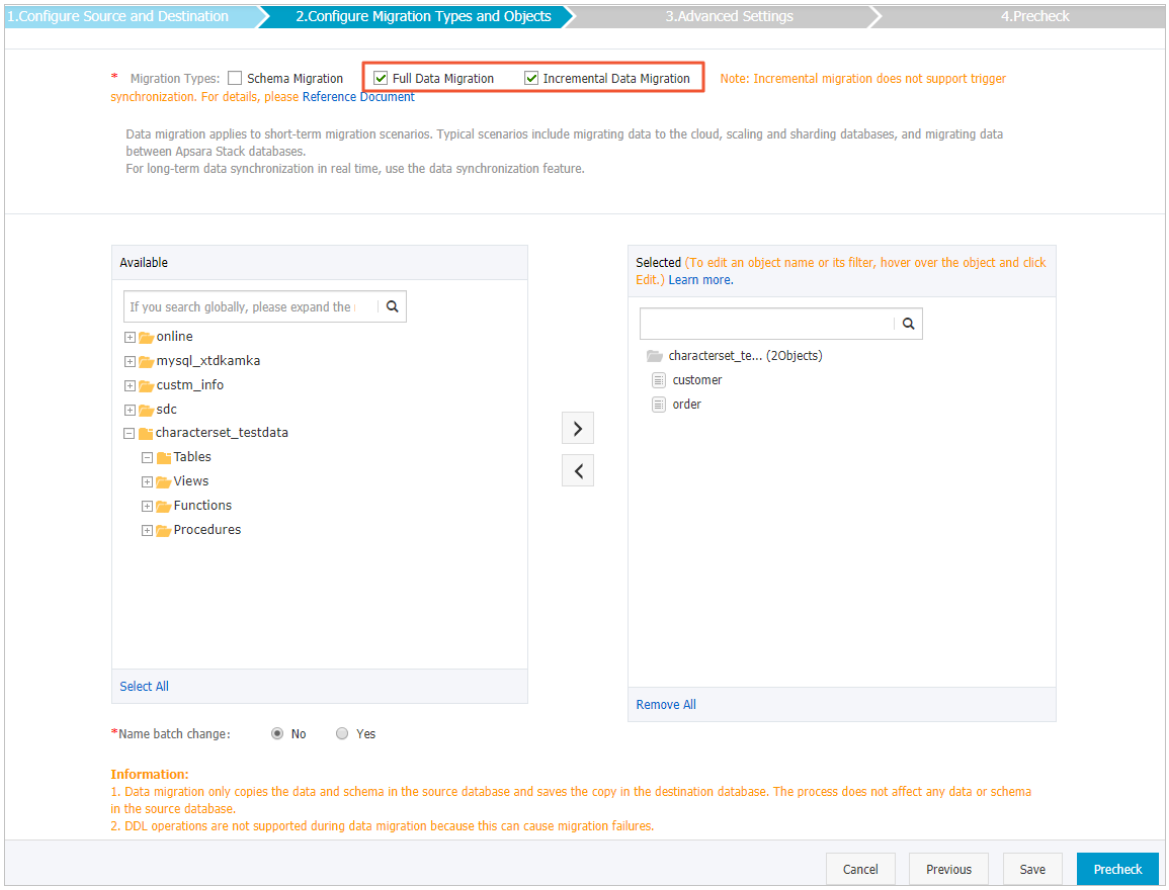
Region	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you use an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **RDS Instance**.
	Instance Region	Select the region where the source RDS instance resides.
	RDS Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. The account must have the read/write permissions on the database to be migrated.
	Database Password	Enter the password of the database account. **Note** After you specify the information about the self-managed Oracle database, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.

Region	Parameter	Description
	Encryption	Select **Non-encrypted** or **SSL-encrypted**. If you want to select **SSL-encrypted**, you must have enabled SSL encryption for the RDS instance. For more information, see Configure SSL encryption for an RDS for MySQL instance. **Note** The **Encryption** parameter is available only for data migration instances that reside in mainland China and Hong Kong (China).
Destination Database	Instance Type	Select **RDS Instance**.
	Instance Region	Select the region where the destination RDS instance resides.
	RDS Instance ID	Select the ID of the destination RDS instance.
	Database Account	Enter the database account of the destination RDS instance. The account must have the read/write permissions on the destination database.
	Database Password	Enter the password of the database account. **Note** After you specify the information about the RDS instance, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.
	Encryption	Select **Non-encrypted** or **SSL-encrypted**. If you want to select **SSL-encrypted**, you must have enabled SSL encryption for the RDS instance. For more information, see Configure SSL encryption for an RDS for MySQL instance. **Note** The **Encryption** parameter is available only for data migration instances that reside in mainland China and Hong Kong (China).

6. In the lower-right corner of the page, click **Set Whitelist and Next**.

   Note The CIDR blocks of DTS servers are automatically added to the whitelist of the source ApsaraDB RDS for MySQL instance. This ensures that DTS servers can connect to the source RDS instance.

7. Select the migration types and objects to be migrated.



Parameter	Description
Migration Types	Schema migration is completed in Step 1. Therefore, you do not need to select Schema Migration in this step. ○ To perform only full data migration, select only **Full Data Migration**. ○ To migrate data with minimal downtime, select **Full Data Migration** and **Incremental Data Migration**. **Note** ○ If **Incremental Data Migration** is not selected, do not write data into the source database during full data migration. This ensures data consistency between the source and destination databases. ○ The following SQL operations can be migrated during incremental data migration: INSERT, UPDATE, DELETE, REPLACE, CREATE FUNCTION, CREATE INDEX, CREATE PROCEDURE, CREATE TABLE, CREATE VIEW, DROP INDEX, DROP TABLE, RENAME TABLE, and TRUNCATE TABLE.

Parameter	Description
Objects	In the **Available** section, select the objects whose schema has been migrated in Step 1, and click the icon to add the objects to the **Selected** section. **Warning** We recommend that you do not use the object name mapping feature. Otherwise, the data migration task fails.

8. Start the data migration task.
   i. In the lower-right corner of the page, click **Precheck**.

   > **Note**
   > - Before you can start the data migration task, a precheck is performed. A data migration task can be started only if it passes the precheck.
   > - If the task fails to pass the precheck, click the icon next to each failed item to view details. Fix the issues as instructed and run the precheck again.

   ii. After the task passes the precheck, click **Next**.
   iii. In the **Confirm Settings** dialog box that appears, specify the **Channel Specification** parameter and select **Data Transmission Service (Pay-As-You-Go) Service Terms**.
   iv. Click **Buy and Start** to start the data migration task.

Step 3: Switch your workloads

Switch your workloads by using one of the following solutions:

- Rollback solution: You do not need to edit the code of your application. However, a rollback failure may occur.
  i. After data migration is completed, verify the data in the destination ApsaraDB RDS for MySQL instance.
  ii. Prepare a rollback solution, and then switch your workloads to the destination instance. The rollback solution is based on a data migration task in the opposite direction. The task allows you to switch workloads back to the original source instance. For more information, see Switch workloads to the destination database.
  iii. Test the features of your business. If the features function as expected, delete the data migration task in the opposite direction.
  iv. (Optional)If you no longer need the source ApsaraDB RDS for MySQL instance, release the instance.
- Dual-write solution: This solution ensures a high success rate of rollback. However, great efforts are required to edit your application code.
  i. Edit the application code to implement the dual-write solution. The dual-write solution writes data changes to both the source and destination instances.
  ii. Stop the data migration task.

- Full data migration

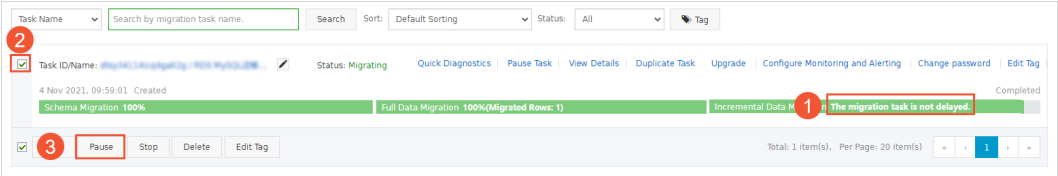
  Do not manually stop a task during full data migration. Otherwise, the system may fail to migrate all data. Wait until the migration task automatically ends.

- Incremental data migration

  The task does not automatically end during incremental data migration. You must manually stop the migration task.

  a. Wait until the task progress bar shows **Incremental Data Migration** and **The migration task is not delayed**. Then, stop writing data to the source database for a few minutes. In some cases, the progress bar shows the delay time of **incremental data migration**.

  b. After the status of **incremental data migration** changes to **The migration task is not delayed**, manually stop the migration task.



> **Note** Make sure that no sessions are performing write operations. To retrieve session information, you can Log on to the source instance and execute the `show processlist;` statement.

iii. Verify the data in the destination ApsaraDB RDS for MySQL instance.

iv. Enable the dual-write solution for your application. Then, data changes will be written to both the source and destination instances.

v. Test the features of your business on the destination instance. If the features function as expected, disable the dual-write solution.

vi. (Optional)If you no longer need the source ApsaraDB RDS for MySQL instance, release the instance.

5.Configure a data synchronization task for a source database that contains a trigger

Synchronizing an entire database that contains a trigger may cause data inconsistency between the source and destination databases if the trigger updates a table. This topic uses an example to describe how to configure a data synchronization task in this scenario.

Background information

A database named triggertestdata contains a trigger and the following two tables: parent and child. If a data record is inserted into the parent table, the trigger inserts the data record into the child table.

> **Note** The following table describes the definitions of the tables and trigger.

Object type	Name	Definition

Table	parent	`CREATE TABLE `parent` (`
		`  `user_vs_id` int(11) NOT NULL AUTO_INCREMENT,`
		`  `name` varchar(30) DEFAULT NULL,`
		`  PRIMARY KEY (`user_vs_id`)`
		`) ENGINE=InnoDB AUTO_INCREMENT=2001 DEFAULT`
		`CHARSET=utf8`

Table	child	`CREATE TABLE `child` (`
		`  `sys_child_id` int(11) NOT NULL AUTO_INCREMENT,`
		`  `user_vs_id` int(11) DEFAULT NULL,`
		`  `name` varchar(30) DEFAULT NULL,`
		`  PRIMARY KEY (`sys_child_id`)`
		`) ENGINE=InnoDB AUTO_INCREMENT=2001 DEFAULT`
		`CHARSET=utf8`

Trigger	data_check	`CREATE  TRIGGER `triggertestdata`.`data_check``
		`AFTER INSERT  ON triggertestdata.parent`
		` FOR EACH ROW`
		`begin`
		`insert into child(user_vs_id, name)`
		`values(new.user_vs_id, new.name) ;`
		`end;`

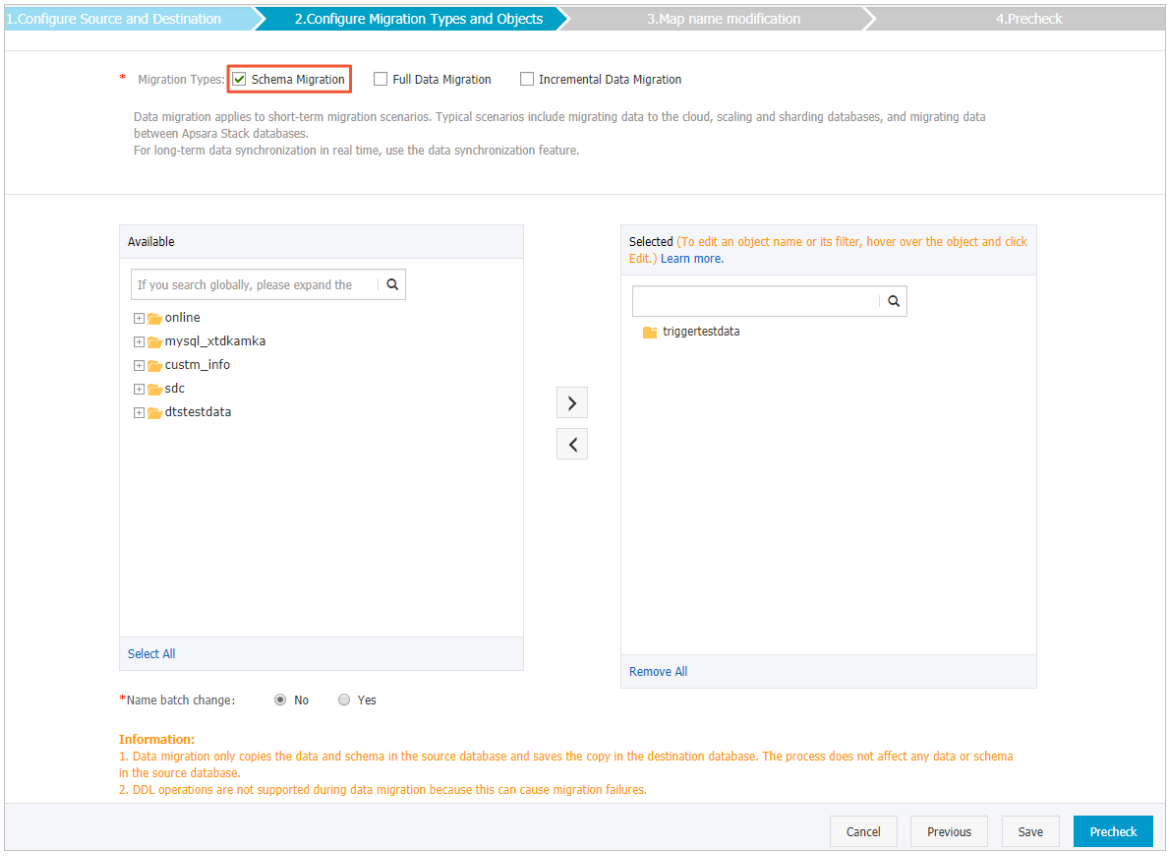
In this example, if an INSERT operation is performed on the parent table during data synchronization, the data in the source child table will be inconsistent with the data in the destination child table. To resolve this issue, you must delete the trigger in the destination database.

Procedure

This section takes data synchronization from a user-created MySQL database to ApsaraDB RDS for MySQL as an example. For more examples, see Overview of data synchronization scenarios.

1. Create a data migration task to migrate the schema of the source database to the destination database. For more information, see Migrate data from a self-managed MySQL database to an ApsaraDB RDS for MySQL instance. In this example, select only **Schema Migration** for the **Migration Types** parameter.

   > **Note** In the **Configure Migration Types and Objects** step, select only **Schema Migration** for the **Migration Types** parameter, and then select databases and tables to be migrated.

2. During schema migration, DTS migrates triggers to the destination database. After the data migration task is completed, log on to the destination database and run the following command to delete the migrated triggers.

   ```
   drop trigger <Trigger name>;
   ```

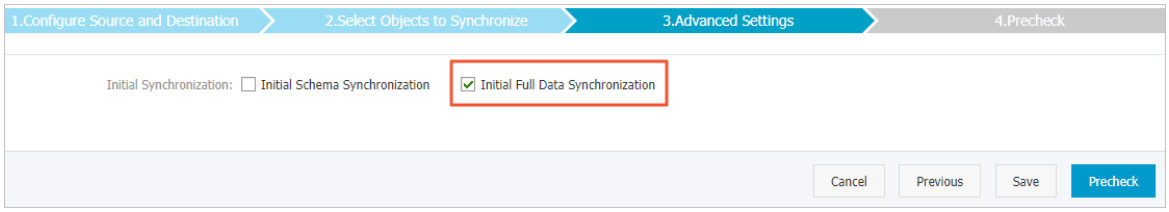
   Example:

   ```
   drop trigger data_check;
   ```

3. Create a data synchronization task to synchronize data from the source database to the

destination database. For more information, see Synchronize data from a self-managed MySQL database hosted on ECS to an ApsaraDB RDS for MySQL instance. In this example, select only **Initial Full Data Synchronization** for the **Initial Synchronization** parameter.

> **Note**
>
> - When you create the data synchronization task, you must select the same objects as those you selected in Step 1.
> - The schema migration has been completed. Therefore, you only need to select **Initial Full Data Synchronization** for the **Initial Synchronization** parameter.



Check data consistency

1. Log on to the source database and insert a data record into the parent table.

   ```
   insert into parent values(1,'testname');
   ```

   After a data record is inserted, the trigger inserts the data record into the source child table.

2. Log on to the source and destination databases. Query the data of the source child table and the destination child table. Check whether the data is consistent between the two tables.
   - Query result of the source child table

	sys_child_id	user_vs_id	name
1	1	1	testname

   - Query result of the destination child table

	sys_child_id	user_vs_id	name
1	1	1	testname

   The results show that the data in the source child table is consistent with the data in the destination child table.

6.Migrate data between databases that have different names

When you configure a data migration task, you can use the object name mapping feature to change the name of the destination database. This feature allows you to migrate data between databases that have different names.

Scenarios

- Data migration between databases that belong to the same instance.
- Data migration between databases that belong to different instances.

Procedure

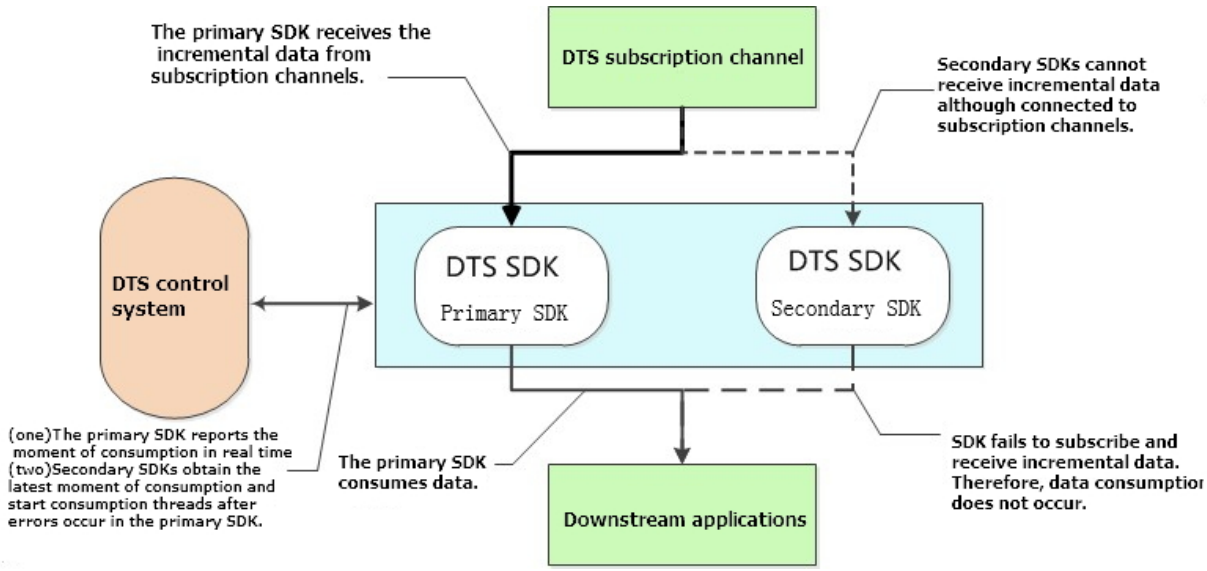
In the **Configure Migration Types and Objects** step, change the name of the destination database by using the object name mapping feature. For more information, see Object name mapping.

7.Disaster tolerance of data subscription SDK

To simplify the complexity of using the SDK, Data Transmission Service (DTS) implements the SDK disaster tolerance mechanism. The mechanism relies on the real-time reporting of consumption time to the DTS central control node when the SDK consumes data. This section introduces how to use the SDK disaster tolerance mechanism provided by DTS.

1. In the code for consuming messages in the SDK, each consumption of a message calls the ackAsConsumed API to report the consumption time to DTS.
2. A subscription channel requires you to start at least a primary SDK and a secondary SDK.

Consumption architecture



Enter the connection type

In a subscription channel of DTS, only one SDK can pull incremental data. If multiple downstream SDKs exist in a subscription channel, only one of them can receive incremental data. Therefore, the SDK disaster tolerance architecture can be set up.

In the sample architecture in the preceding figure, two downstream SDKs are connected in one subscription channel. The two SDKs serve as a primary and secondary SDK for each other. Within the same period of time, only the primary SDK can subscribe to and consume the incremental data. In case of primary SDK exceptions or network connection exceptions, DTS initiates automatic failover to the secondary SDK and starts the secondary SDK using the last consumption time. Every time the primary SDK consumes a message, it reports an ACK message to the DTS control system, which means it reports the consumption time to the control system.

8.Set a cache update policy by using the change tracking feature

This topic describes how to set a cache update policy for a MongoDB or Redis database when you use the change tracking feature. The cache update policy allows you to achieve high reliability and low latency.

Prerequisites

A change tracking task (the previous version) is configured. For more information, see Track data changes from a PolarDB-XPolarDB-X 1.0 instance.

Context

To accelerate business access and improve concurrent read performance, you can build a caching layer in your business architecture. The read requests are routed to the caching layer. The memory reading mechanism of the caching layer allows you to improve the read performance of your business. To ensure data integrity, the updated business data is persistently stored in ApsaraDB RDS for MySQL. In this case, you can set a cache update policy to ensure that the cached data is updated when the business data is updated.

Before you begin

Create an AccessKey pair, which consists of an AccessKey ID and AccessKey secret. For more information, see Create an AccessKey pair.

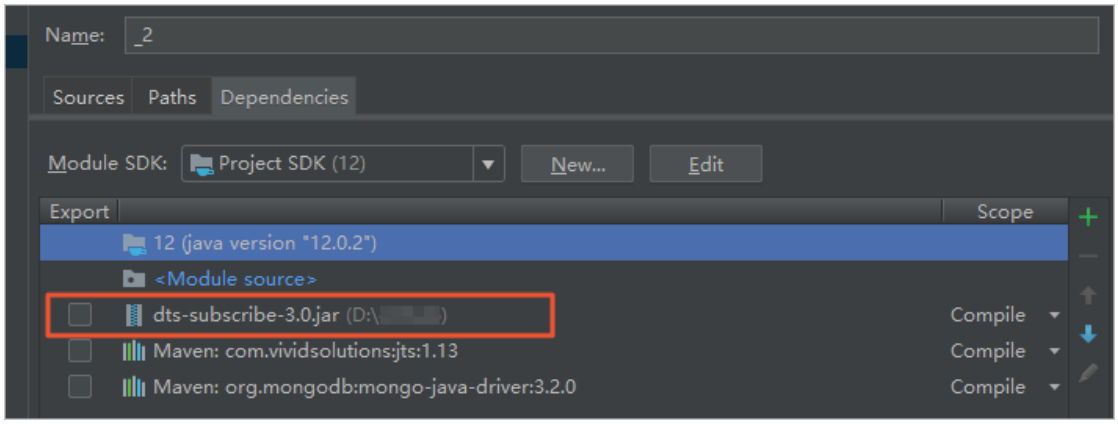
> **Notice** If you track and consume data as a RAM user, the **AliyunDTSFullAccess** permission must be granted to the RAM user. For more information, see Use a system policy to authorize a RAM user to manage DTS instances.

Procedure

IntelliJ IDEA (Community Edition 2018.1.4 Windows) is used in this example.

1. Download the SDK package (dts-subscribe-3.0.jar).
2. Add the *dts-subscribe-3.0.jar* package to the project dependencies.
   i. Open IntelliJ IDEA, choose **File > New > Project**, and then create a Maven project.
   ii. In the menu bar, choose **File > Project Structure**.
   iii. In the left-side navigation pane, click **Modules**. On the right side of the page, choose  **> 1 JARs or directories**.

iv. Select the *dts-subscribe-3.0.jar* package and add it to the project dependencies.



3. Perform the following steps to set a cache update policy for a MongoDB or Redis database:
   - Run the demo code for MongoDB
     a. Double-click the *pom.xml* file and add the following settings to the file:

```
<dependencies>
    <dependency>
        <groupId>com.vividsolutions</groupId>
        <artifactId>jts</artifactId>
        <version>1.13</version>
    </dependency>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>3.2.0</version>
    </dependency>
</dependencies>
```

     b. Save the settings and wait for the dependencies to be loaded.

> **Note** We recommend that you enable IntelliJ IDEA to update Maven dependencies.

     c. Create a Java class file named *DTSMySQL2Mongo* in the Maven project.
     d. Download the demo code for MongoDB.
     e. Copy the downloaded demo code and replace the existing content in the *DTSMySQL2Mongo* file with the demo code.

f. Edit the code in the *DTSMySQL2Mongo* file and set the parameters.

Parameter	Description
accessKey	Enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account. For more information, see Before you begin.
accessSecret	
Subscription_Instance_ID	Enter the ID of the change tracking instance. To obtain the ID of the change tracking instance, perform the following steps: Log on to the DTS console. In the left-side navigation pane, click **Change Tracking**.
mongUrl	Enter the connection string of the MongoDB database and the name of the authentication database. The format is `<Connection string>:<Service port number>/<Authentication database name>`, for example, `ds-bp********-pub.mongodb.rds.aliyuncs.com:3717/admin`.
mongUserName	Enter the database account of the MongoDB database. **Note** The database account must have the read and write permissions on the destination database. For example, if the tracked data is written to a MongoDB database named testdata, the database account must have the read and write permissions on the testdata database.
mongUserPassword	Enter the password of the database account.

g. In the top menu bar of IntelliJ IDEA, choose **Run > Run** to run the program.

- Run the demo code for Redis

a. Double-click the *pom.xml* file and add the following settings to the file:

```
<dependencies>
    <dependency>
        <groupId>com.vividsolutions</groupId>
        <artifactId>jts</artifactId>
        <version>1.13</version>
    </dependency>
    <dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
        <version>2.7.2</version>
    </dependency>
</dependencies>
```

b. Save the settings and wait for the dependencies to be loaded.

> **Note** We recommend that you enable IntelliJ IDEA to update Maven dependencies.

c. Create a Java class file named *DTSMySQL2Redis* in the Maven project.

d. Download the demo code for Redis.

e. Copy the downloaded demo code and replace the existing content in the *DTSMySQL2Redis* file with the demo code.

f. Edit the code in the *DTSMySQL2Redis* file and set the parameters.

```
public class DTSMySQL2Redis {

    private static final String accessKey = "";
    private static final String accessSecret = "";
    private static final String Subscription_Instance_ID = "dts";
    private static String redisPassword = "";
    private static int redisPort = 6379;

    private static String redisUrl = "r-redis.rds.aliyuncs.com";
```

Parameter	Description
accessKey	Enter the AccessKey ID and AccessKey secret of the Alibaba Cloud account. For more information, see Before you begin.
accessSecret	
Subscription_Instance_ID	Enter the ID of the change tracking instance. To obtain the ID of the change tracking instance, perform the following steps: Log on to the DTS console. In the left-side navigation pane, click **Change Tracking**.
redisPassword	Enter the database password of the Redis database. **Note** If a password is set for the Redis database, you must change the code in line 73 from `pool = new JedisPool(config, redisUrl, redisPort, 3000);` to `pool = new JedisPool(config, redisUrl, redisPort, 3000,redisPassword)`.
redisPort	Enter the service port number of the Redis database. The default port number is 6379.
redisUrl	Enter the endpoint (domain name or IP address) of the Redis database.

g. In the top menu bar of IntelliJ IDEA, choose **Run > Run** to run the program.

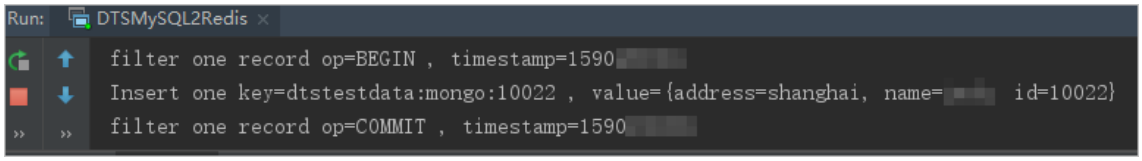
4. When the program runs, it retrieves incremental data from the source database and writes the data to the MongoDB or Redis database.

Result

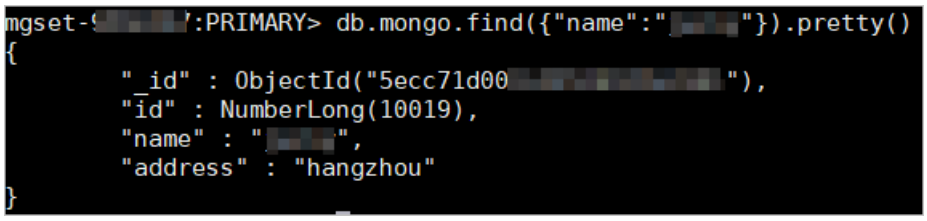
1. Log on to the source database and insert several data records into the source table. Then, the program tracks the inserted data records and writes the data records to the destination database.
   - Running result of the demo code for MongoDB

```
Run:  DTSMySQL2Mongo
Insert one document=Document{{id=10019, name=, address=hangzhou}}
recive one message count=1
filter one record op=COMMIT , timestamp=159
```

   - Running result of the demo code for Redis



2. Log on to the destination database and query the inserted data records. Then, the data records are written to the database.
   - Query result for the MongoDB database



   - Query result for the Redis database